Theoretical Computer Science 410 (2009) 1932–1941

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Efficient probability amplification in two-way quantum finite automata

Abuzer Yakaryılmaz *, A.C. Cem Say

Boğaziçi University, Department of Computer Engineering, Bebek 34342 İstanbul, Turkey

ARTICLE INFO

Keywords:
Two-way quantum finite automata
Probability amplification

ABSTRACT

In classical computation, one only needs to sequence $O(\log \frac{1}{\epsilon})$ identical copies of a given probabilistic automaton with one-sided error $p < 1$ to run on the same input in order to obtain a two-way machine with error bound ϵ. For two-way quantum finite automata (2qfa's), this straightforward approach does not yield efficient results; the number of machine copies required to reduce the error to ϵ can be as high as $(\frac{1}{\epsilon})^2$. In their celebrated proof that 2qfa's can recognize the non-regular language $L = \{a^n b^n \mid n > 0\}$, Kondacs and Watrous use a different probability amplification method, which yields machines with $O((\frac{1}{\epsilon})^2)$ states, and with runtime $O(\frac{1}{\epsilon}|w|)$, where w is the input string. In this paper, we examine significantly more efficient techniques of probability amplification. One of our methods produces machines which decide L in $O(|w|)$ time (i.e. the running time does not depend on the error bound) and which have $O((\frac{1}{\epsilon})^{\frac{2}{c}})$ states for any given constant $c > 1$. Other methods, yielding machines whose state complexities are polylogarithmic in $\frac{1}{\epsilon}$, including one which halts in $o(\log(\frac{1}{\epsilon})|w|)$ time, are also presented.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

Researchers considering restricted theoretical models of quantum computers have focused on the quantum counterparts of classical finite state automata [1–6,9–12,14]. It is well known [13] that two-way deterministic finite automata recognize all and only the regular languages. There exist two-way probabilistic finite state automata that can recognize some non-regular languages (like $L = \{a^n b^n | \ n > 0\}$) [8]; however, these machines necessarily have exponential running times [7]. In 1997, Kondacs and Watrous [10] introduced two-way quantum finite automata (2qfa's), and proved them to be more powerful than their classical counterparts by describing a method for constructing 2qfa's that recognize L in time linear in terms of the input length for any given (one-sided) error bound $\epsilon > 0$. Machines built according to this method have $O((\frac{1}{\epsilon})^2)$ states, and they halt after $O(\frac{1}{\epsilon}|w|)$ steps, where w is the input string.

The conventional way of recognizing a language with desired error bound ϵ when given a machine for that language with one-sided error, say, $\frac{1}{2}$, involves a procedure where the computation is repeated $O(\log \frac{1}{\epsilon})$ times. In the two-way finite automata setup, this repetition can be achieved by just sequencing $O(\log \frac{1}{\epsilon})$ copies of the given machine to process the input one after another, so that the possibly erroneous response type is produced by the resulting automaton only if it is produced by all the copies in the sequence. Viewed in this context, the dependences of the runtime and state complexity functions of the machines of [10] on $\frac{1}{\epsilon}$ are surprisingly great. In this paper, we examine significantly more efficient techniques of probability amplification, and apply them to obtain faster and smaller 2qfa's that recognize L with error bound ϵ.

Machines built according to the method of Kondacs and Watrous compare the numbers of a's and b's by utilizing the positions of the tape heads in different computation paths. Amplification of the success probability to the desired level is achieved by introducing more and more parallel computation paths, which traverse the input string with lower and lower speeds, increasing the runtime. By first checking whether the numbers of a's and b's are equivalent modulo a divisor k, we

* Corresponding author.
E-mail addresses: abuzer@boun.edu.tr (A. Yakaryılmaz), say@boun.edu.tr (A.C.C. Say).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.01.029

modify (Section 4) the Kondacs–Watrous method to then make a correspondingly faster comparison between the quotients obtained when the numbers of a's and b's are divided by k. Choosing k and the number of computation paths carefully, we are able to produce machines that halt in $O(|w|)$ time for any ϵ, removing the dependence on the error bound entirely. This is the best runtime possible, because one cannot even read the complete string w in time less than that. In our method, only the size of the constructed program is dependent on ϵ.

In Section 5, we examine ways of reducing the state cost of probability amplification, employing generalizations of the quotient comparison method of Section 4. We present a method which can be used to construct machines with runtime $O(|w|)$, and state set size $O((\frac{1}{\epsilon})^{\frac{2}{c}})$ for any given constant $c > 1$. We also show that two other constructions with even smaller state complexities exist, if one allows the runtime of the resulting machines to have small dependences on ϵ. One of these methods yields machines with $O(\frac{\log^3 \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}})$ states and runtime $O(\frac{\log \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}}|w|)$, which is still better than that of the conventional probability amplification technique mentioned above. The other method produces machines with the lowest state complexity ($O(\log^2(\frac{1}{\epsilon})\log\log \frac{1}{\epsilon})$) known so far for 2qfa's that recognize L in time linear in terms of the input length, and with runtime $O(\log(\frac{1}{\epsilon})|w|)$.

All the techniques described in Section 5 involve the repeated application of the quotient comparison procedure in multiple passes over the input string, with the aim of reducing the error probability to the desired value. As noted by Watrous [14], a naive approach which cascades k copies of a QFT-based 2qfa with error $\frac{1}{N}$ ends up with an error probability which is considerably worse than N^{-k}. The reason is that the basic algorithm depends on different computation paths reaching the end of the string at the same time just if $w \in L$, and unwanted interference occurs when, for instance, a path which takes i steps in the first pass and j steps in the second pass reaches the end of the second pass at exactly the same time as another path which takes j steps in the first pass and i steps in the second pass, even when $w \notin L$. In our framework, the quotient comparison passes are performed with different divisors in each round, and these numbers are chosen in a way which guarantees that such undesired collisions never occur.

The remainder of this paper is organized as follows. Section 2 is a brief review of two-way quantum finite automata. In Section 3, we describe the approaches taken by other researchers to build 2qfa's for recognizing L. In Sections 4 and 5, we present 2qfa constructions for L embodying our methods of probability amplification. We conclude with Section 6.

2. Two-way quantum finite state automata

We consider two-way quantum finite state automata (2qfa's) as defined in [10].

Like their classical counterparts, 2qfa's have finite state controls and read-only input tapes with two-way tape heads. Formally, a 2qfa is a 6-tuple $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$. Q is a finite set of states, $q_0 \in Q$ is the initial state, $Q_{acc} \subset Q$ is the set of accepting states, and $Q_{rej} \subset Q$ is the set of rejecting states. Q_{acc} and Q_{rej} are disjoint, and their union is the set of halting states. The elements of $Q_{non} = Q \setminus (Q_{acc} \cup Q_{rej})$ are non-halting states. Σ is the input alphabet. The symbols $\{¢, \$\} \notin \Sigma$ are used to mark the left and right ends of the input string, respectively. $\Gamma = \Sigma \cup \{¢, \$\}$ is the tape alphabet. δ is the transition function, which governs the behavior of M, as will be described below.

In all 2qfa's described in this paper, every transition entering the same state involves the tape head moving in the same direction (left, right, or stationary). We represent this feature of a state q using the appropriate one of the notations $\overleftarrow{q}$, $\overrightarrow{q}$, or $\downarrow q$ for this state in the machine description. With this simplification, considering the Hilbert space $\ell_2(Q)$, a syntactically correct 2qfa can be specified easily by just providing a unitary operator $V_\sigma : \ell_2(Q) \to \ell_2(Q)$ for each symbol $\sigma \in \Gamma$. More formally,

$$\delta(q, \sigma, q', d_{q'}) = \langle q'|V_\sigma|q\rangle$$

is the amplitude with which the machine currently in state q and scanning symbol σ will jump to state q' and move the head in direction $d_{q'}$. Here, $d_{q'} \in \{-1, 0, 1\}$ is the direction of the tape head determined by q'. For the remaining directions, all transitions with target q' have amplitude zero.

The configurations of a 2qfa are pairs of the form (*state*, *head position*). Note that δ may allow a single configuration to have multiple successors with non-zero amplitude. It may also be the case that a single configuration has multiple predecessors. Just as in classical nondeterministic or probabilistic models, the computation of a 2qfa on an input string can be represented by the directed graph of its configurations, with edges corresponding to transitions from non-halting predecessors to successors. Initially, the head is on the left end-marker, and so the machine starts in the configuration $(q_0, 0)$. A *computation path* is defined to be any path in the aforementioned computation graph which starts from $(q_0, 0)$. At later steps of the computation, due to its quantum nature, the machine may exist in superpositions of more than one configuration. It is sometimes useful to visualize such superpositions of multiple configurations as snapshots of the machine running in multiple parallel computation paths or branches. The amplitude of a configuration c in such a superposition is determined by the amplitudes of its non-halting predecessors in the superposition of the previous step, and the amplitudes of the transitions to c, in a manner analogous to classical probabilistic computation. However, since these amplitudes are complex numbers, parallel computation paths may interfere with each other in ways quite unlike what can happen with classical machines. Clearly, for two computation paths to be able to interfere, that is, to merge in the same configuration

in the same step, their lengths must be equal. Most of the analysis in the remaining sections of this paper will focus on the relative lengths of such paths.

In each step of its execution, a 2qfa undergoes two linear operations. The first one is a unitary transformation of the current superposition according to δ, and the second one is a measurement. The observable describing this measurement process is designed so that the outcome of any observation is "accept", "reject", or "non-halting". The probability of each outcome is determined by the amplitudes of the relevant configurations in the present superposition. The contribution of each configuration to this probability is the modulus squared of its amplitude. For instance, the outcome "accept" will be measured with probability $\sum_{c\in Q_{acc}\times\mathbb{Z}_n}|\alpha_c|^2$, where α_c is the amplitude of configuration c, and n is the length of the tape. If we measure "accept" or "reject", the computation halts. (This is why we were able to state that halting configurations have no successors in the preceding discussion about the computation graph.) If we measure "non-halting", the machine continues running from a superposition of the non-halting configurations, obtained by normalizing the projection of the superposition before the measurement onto $\text{span}\{|c\rangle | c \in Q_{non}\times\mathbb{Z}_n\}$.

A 2qfa M is said to recognize a language A with error bounded by ϵ if M's computation results in "accept" being measured for all members of A with probability at least $1-\epsilon$, and "reject" being measured for all other inputs with probability at least $1-\epsilon$.

For a more detailed coverage of the technical properties of 2qfa's, we refer the reader to [10].

3. Previous work

3.1. N-way branching with a single pass

In their seminal paper [10], Kondacs and Watrous presented a method for constructing a 2qfa that recognizes $L = \{a^n b^n |\ n > 0\}$ for a given positive one-sided error bound ϵ. According to this construction, the machine first attempts to validate that the input string is of the form $\{a^m b^n |\ m, n > 0\}$, rejecting and halting otherwise. If this first stage is completed without rejection, it leaves the tape head over the right end-marker \$. The second and final stage begins with the computation branching into $N = \lceil \frac{1}{\epsilon} \rceil$ paths, each with amplitude $\frac{1}{\sqrt{N}}$. In each of these paths, the head travels with different speeds on a's and b's towards the left end-marker. In path P_j $(1 \le j \le N)$ the head waits j steps over each a and $N-j+1$ steps over each b before moving left. These waiting times have been selected to ensure that all paths will finish the traversal of the string $a^m b^n$ and arrive at the left end-marker simultaneously only if $m = n$, and no two paths will arrive at ¢ at the same time if $m \neq n$. When the tape head arrives at the end symbol ¢, each path performs the Quantum Fourier Transform (QFT) to a common set of N target states. Since the QFT is an essential ingredient of all the algorithms that will be discussed in this paper, we will describe its use in some detail here.

The N-way QFT is the transformation

$$V_\sigma |s_j\rangle = \frac{1}{\sqrt{N}} \sum_{l=1}^{N} (e^{\frac{2\pi i}{N} jl}) |t_l\rangle, \quad 1 \le j \le N, \tag{1}$$

which involves two sets of N states, that we will call the *source states* (the s_j) and the *target states* (the t_l), respectively. t_N is called the *distinguished target state*. Let the list $(d_{t_1}, d_{t_2}, \ldots, d_{t_N})$ contain the head directions (represented by values from the set $\{-1, 0, 1\}$) associated with the target states. Assume that the symbol at position p of the tape is σ. We will consider two cases in our examination of the use of the QFT.

In the first case, only one of the source states, say, s_i, appears in a configuration (s_i, p) with non-zero amplitude α in the current superposition of the machine. After the application of the transition function, the modulus of the amplitude of each of the N configurations $(t_1, p + d_{t_1}), (t_2, p + d_{t_2}), \ldots, (t_N, p + d_{t_N})$ will be $\frac{|\alpha|}{\sqrt{N}}$; that is, each configuration will contribute equally to the observation probabilities as any of the others. (A configuration (q, p) can have some of the target states in its successors with non-zero amplitude only if q is one of the source states, due to the unitarity of V_σ.) To summarize, if a single computation path undergoes a particular QFT at any step, it branches to N paths, among which its probability is divided equally.

In the second case, assume that each of the configurations (s_i, p) $(1 \le i \le N)$ has amplitude α in the current superposition. After the QFT, these give rise to

$$\frac{\alpha}{\sqrt{N}} \sum_{j=1}^{N} \sum_{l=1}^{N} e^{\frac{2\pi i}{N} jl} |t_l, p + d_{t_l}\rangle = \alpha\sqrt{N} |t_N, p + d_{t_N}\rangle, \tag{2}$$

with probability contribution $|\alpha|^2 N$. All the remaining target configurations $((t_1, p+d_{t_1}), (t_2, p+d_{t_2}), \ldots, (t_{N-1}, p+d_{t_{N-1}}))$ that we saw in the first case have amplitude zero. To paraphrase, if N parallel computation paths with equal amplitude undergo the QFT at the same step, they merge into a single configuration containing the distinguished target state, which inherits all of their combined probability.

Returning to the description of the Kondacs–Watrous construction, we now specify the target states of the QFT performed on the left end-marker by the N computation paths. The distinguished target state is an accept state. The remaining $N-1$ target states are reject states.

If the numbers of a's and b's are not equal, each computation path arrives at ¢ at a different time, and therefore branches to the accept state with probability $\frac{1}{N}$, and to some reject state with probability $1-\frac{1}{N}$. Therefore, the overall acceptance probability is also $\frac{1}{N}$, and the overall rejection probability is $1-\frac{1}{N}$. If the numbers of a's and b's are equal, all paths perform the QFT at the same time, and the machine accepts with probability 1.

As is evident from this description, the reduction of the error to a lower level is achieved by the introduction of more parallel paths, traversing the tape with slower speeds. This reflects on both the time and state complexities; for error bound ϵ and input string w, a machine built according to this method has $O((\frac{1}{\epsilon})^2)$ states, and halts within $O(\frac{1}{\epsilon}|w|)$ steps.

3.2. Two-way branching in multiple passes

Watrous [14] presents an alternative approach to recognize L. After making sure that the input is of the form $\{a^m b^n | \, m, n > 0\}$, the head moves to the last a. At that point, the computation branches to two paths. One path starts on the first b and walks over the b's to the right end-marker, while the other path starts on the last a and walks over the a's (with the same speed as the first one) to the left end-marker. In both paths, the head then returns to the "border" between the a's and the b's. Upon crossing the border, both paths perform a QFT with $N = 2$ to the same target state set. If the two targets of this QFT were both halting states, as described in the previous subsection, this machine would clearly have a fixed (one-sided) error bound of $\frac{1}{2}$. In order to decrease this error, the first target is set to be a reject state, whereas the distinguished target is made a non-halting state. The computation splits again into two in this non-halting state, and the process described above is repeated many times, with the final round ending with a QFT whose distinguished target is an accept state, and the other target is a reject state. This is a direct application of conventional probability amplification by repetition.

For members of L, the computation will clearly survive all the QFT's without reaching any reject state, and the input will be accepted with probability 1. For strings of the form $a^m b^n$ $(m \neq n)$, the computation can avoid the reject states of the previous rounds to reach the kth round with a probability of approximately $\sqrt{\frac{1}{\pi(k-1)}}$, rather than the much lower $(\frac{1}{2})^{k-1}$ that one would have in the classical case. The reason of this inefficiency is the fact that different computation paths with the same length may interfere even during the processing of non-members of L. For example, the path which walks over the a's in the first round and the b's in the second round would reach the second QFT at exactly the same time as the path which walks over the b's in the first round and the a's in the second round, and they would therefore avoid the reject state, merging into a single non-halting configuration. In general, any computation path about to enter the kth QFT will have walked over the a's in j rounds $(0 \leq j \leq k)$, and the b's in $k-j$ rounds. For a particular j, there are $\binom{k}{j}$ different orderings of such walks, and all these paths will interfere at the kth QFT at the latest. As a result, the number of rounds required to decrease the erroneous acceptance probability for non-members of L to below a desired positive value ϵ turns out to be $O((\frac{1}{\epsilon})^2)$, exponentially worse than what one would need in an analogous classical setup without such interference. Therefore, for input string w, machines built according to this method have $O((\frac{1}{\epsilon})^2)$ states, and halt within $O((\frac{1}{\epsilon})^2|w|)$ steps.

3.3. Two-way branching in multiple passes with collision avoidance

Atak [4] proposed modifying Watrous' algorithm of Section 3.2 to prevent the unwanted interference of computation paths by incorporating additional waiting states which cause the head to move slower and slower in each new round to every round except the first. We formalized the setup used by Atak to calculate the number of waiting steps that his method would require for any desired value of ϵ. If the number of additional waiting steps in each round of a p-round machine is taken from the list $W = (0, 1, 2^2-2, \ldots, 2^{p-1}-2)$ (where $p > 2$), then no unwanted "collisions" of the kind mentioned above can occur. This leads to a distinct advantage over the method of Section 3.2: the erroneous acceptance probability ϵ decreases from $\sqrt{\frac{1}{\pi p}}$ to $\frac{1}{2^p}$. (See Section 5 for a detailed analysis of a similar issue.) Machines built according to this method have $O(\frac{1}{\epsilon})$ states, and they halt in $O(\frac{1}{\epsilon}|w|)$ steps for input string w. This approach yields more succinct machines than [10] and [14].

4. The quotient comparison method

In this section, we describe a method to construct 2qfa's which recognize L, and whose runtimes do not depend on the error bound ϵ.

For each integer $N > 1$ and $k > 1$, we define $M_{k,N} = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ as follows:

$$
\begin{aligned}
Q \;=\; & \{\overrightarrow{q}_0, \overleftarrow{q}_1, \overrightarrow{q}_2, \downarrow q_3\} \cup \{\overleftarrow{m}_i \mid 0 \leq i \leq k-1\} \cup \{\downarrow r_i \mid 1 \leq i \leq k-1\}, \\
& \cup \{\overrightarrow{m}_{j,i} \mid 1 \leq j \leq N, 0 \leq i \leq k-1\}, \cup \{\downarrow w_{j,l} \mid 1 \leq j \leq N, 1 \leq l \leq \max(j, N-j+1)\}, \\
& \cup \{\downarrow s_i \mid 1 \leq i \leq N\}. \\
Q_{acc} = \{\downarrow s_N\}, \quad & \text{and} \quad Q_{rej} = \{\downarrow q_3\} \cup \{\downarrow r_i | 1 \leq i \leq k-1\} \cup \{\downarrow s_j | 1 \leq j \leq N-1\}. \\
\Sigma = \{a, b\}. &
\end{aligned}
$$

Let each V_σ act as indicated in Fig. 1, and extend each to be unitary. Let δ be related to the V_σ as described in Section 2. Fig. 2 is a high level description of the working of the machine.

Stage 1	Stage 2
$V_{¢}\lvert\overrightarrow{q}_0\rangle = \lvert\overrightarrow{q}_0\rangle,$	$V_{¢}\lvert\overleftarrow{m}_i\rangle = \lvert\downarrow r_i\rangle,\ \ 1 \le i \le k-1,$
$V_{¢}\lvert\overleftarrow{q}_1\rangle = \lvert\downarrow q_3\rangle,$	$V_a\lvert\overleftarrow{m}_i\rangle = \lvert\overleftarrow{m}_{i-1 \bmod (k)}\rangle,\ \ 0 \le i \le k-1,$
$V_a\lvert\overrightarrow{q}_0\rangle = \lvert\overrightarrow{q}_0\rangle,$	$V_b\lvert\overleftarrow{m}_i\rangle = \lvert\overleftarrow{m}_{i+1 \bmod (k)}\rangle,\ \ 0 \le i \le k-1,$
$V_a\lvert\overleftarrow{q}_1\rangle = \lvert\overrightarrow{q}_2\rangle,$	
$V_a\lvert\overrightarrow{q}_2\rangle = \lvert\downarrow q_3\rangle,$	
$V_b\lvert\overrightarrow{q}_0\rangle = \lvert\overleftarrow{q}_1\rangle,$	
$V_b\lvert\overrightarrow{q}_2\rangle = \lvert\overrightarrow{q}_2\rangle,$	
$V_{\$}\lvert\overrightarrow{q}_0\rangle = \lvert\downarrow q_3\rangle,$	
$V_{\$}\lvert\overrightarrow{q}_2\rangle = \lvert\overleftarrow{m}_0\rangle,$	
Stage 3	
$V_{¢}\lvert\overleftarrow{m}_0\rangle = \frac{1}{\sqrt{N}}\sum_{j=1}^{N}\lvert\overrightarrow{m}_{j,0}\rangle,$	
$V_a\lvert\overrightarrow{m}_{j,i}\rangle = \lvert\overrightarrow{m}_{j,i+1}\rangle,\ \ 1 \le j \le N, 0 \le i \le k-2,$	
$V_a\lvert\overrightarrow{m}_{j,k-1}\rangle = \lvert\downarrow w_{j,1}\rangle,\ \ 1 \le j \le N,$	
$V_a\lvert\downarrow w_{j,i}\rangle = \lvert\downarrow w_{j,i+1}\rangle,\ \ 1 \le j < N, 1 \le i \le N-j,$	
$V_a\lvert\downarrow w_{j,N-j+1}\rangle = \lvert\overrightarrow{m}_{j,0}\rangle,\ \ 1 \le j \le N,$	
$V_b\lvert\overrightarrow{m}_{j,i}\rangle = \lvert\overrightarrow{m}_{j,i-1}\rangle,\ \ 1 \le j \le N, 1 \le i \le k-1,$	
$V_b\lvert\overrightarrow{m}_{j,0}\rangle = \lvert\downarrow w_{j,1}\rangle,\ \ 1 \le j \le N,$	
$V_b\lvert\downarrow w_{j,i}\rangle = \lvert\downarrow w_{j,i+1}\rangle,\ \ 1 < j \le N, 1 \le i \le j-1,$	
$V_b\lvert\downarrow w_{j,j}\rangle = \lvert\overrightarrow{m}_{j,k-1}\rangle,\ \ 1 \le j \le N,$	
$V_{\$}\lvert\overrightarrow{m}_{j,0}\rangle = \frac{1}{\sqrt{N}}\sum_{l=1}^{N}(e^{\frac{2\pi i}{N}jl})\lvert\downarrow s_l\rangle,\ \ 1 \le j \le N.$	

Fig. 1. Specification of the transition function of $M_{k,N}$.

Proposition 1. *The 2qfa $M_{k,N}$ recognizes the language $L = \{a^n b^n \mid n > 0\}$ with one-sided error bound $\frac{1}{N}$ for any k. If $k = N$, then the machine halts in $O(|w|)$ steps, and the size of its state set is $O(N^2)$.*

Proof. We will analyze the behavior of the machine. For simplicity, we divide the machine into three stages (as seen in Figs. 1 and 2).

States q_0 through q_3 check the membership of the input in $\{a^m b^n \mid m, n > 0\}$, in Stage 1. States m_0 through m_{k-1} and r_1 through r_{k-1} check whether $m \equiv n \pmod{k}$, in Stage 2. Note that the quantum nature of the machine requires $k-1$ different reject states to be used in this stage, in contrast to the single reject state that is sufficient in a classical machine.

At the beginning of Stage 3, if the machine is still running, we know that $m \equiv n \pmod{k}$, so m and n can be written as $m = xk + c$ and $n = yk + c$, where $0 \le c \le k-1$. We will use the fact that $x = y$ if and only if $m = n$, and therefore a comparison of the smaller values of x and y is sufficient to determine membership of the input in L.

As is evident from Fig. 1, in each computation path P_j in Stage 3 we have a counter, where the machine would be in state $m_{j,i}$ only if the difference between the numbers of a's and b's scanned so far in this pass of the tape is $i \pmod{k}$. What distinguishes this from Stage 2 is that the head pauses in its traversal of the input string and remains stationary for a number of steps specified in Fig. 2 every time the counter is set back to zero. When walking over the a's, the machine enters these waiting states at the squares numbered $k, 2k, \ldots, xk$ ($\lfloor \frac{m}{k} \rfloor = x$ times). While traversing the b's, the machine enters the

1 Scan the input from ¢ to \$ to check whether it is of the form $a^m b^n$; $m, n > 0$. If not, REJECT.

2 Scan the input from \$ to ¢ to check whether $m \equiv n \mod (k)$. If not, REJECT.

3 While on ¢, branch the computation to N different paths (with amplitude $\frac{1}{\sqrt{N}}$). For each computation path P_j $(1 \leq j \leq N)$, scan the input from ¢ to \$ as follows:

While scanning the a's, enter an $(N - j + 1)$–step waiting cycle at every k^{th} symbol (totally $\lfloor \frac{m}{k} \rfloor$ times), counting the a's modulo k simultaneously. Let c be the value of this count at the end of the a region.

While scanning the b's, enter a j–step waiting cycle at the $(c+1)^{st}$ b, and at every k^{th} b after that (totally $\lfloor \frac{n}{k} \rfloor$ times).

Upon reaching \$, perform an N–way QFT to a common target state set. The distinguished target is an accept state, and the other targets are reject states.

Fig. 2. Description of the quotient comparison method.

waiting states at the squares numbered $m + c + 1 + 0k$, $m + c + 1 + 1k$, ..., $m + c + 1 + (y - 1)k$ ($\lfloor \frac{n}{k} \rfloor = y$ times). Upon reaching the right end-marker, each path P_j will be in the state $m_{j,0}$, since it is known that $m \equiv n \mod (k)$.

At this point, each path performs the QFT. As described in Section 3.1, if all paths make the QFT at the same time, rejecting configurations are canceled out, the paths merge on the accept configuration, and so the string is accepted with probability 1. If all paths make the QFT at different times, then each path ends by accepting with probability $\frac{1}{N}$, and rejecting with probability $1 - \frac{1}{N}$, and therefore the overall acceptance and rejection probabilities of the input are also $\frac{1}{N}$, and $1 - \frac{1}{N}$, respectively.

The number of steps taken by path P_j in Stage 3 is

$$steps_j = |w| + xN + (y - x)j + x + 1. \tag{3}$$

If $m = n$, then $x = y$, and $steps_j = |w| + xN + x + 1$. Since the number of steps in this case is independent of j, all computation paths make the QFT simultaneously. Therefore, any member of L is accepted with probability 1.

If $m \neq n$, then $x \neq y$, and the length of path P_j depends on j. If the lengths of paths P_i and P_j are equal, we have

$$\begin{aligned} steps_i &= steps_j \\ |w| + xN + (y - x)i + x + 1 &= |w| + xN + (y - x)j + x + 1 \\ (y - x)i &= (y - x)j \end{aligned}$$

which is true only if $i = j$ since $y - x \neq 0$, so all different paths make the QFT in different times when the input is of the form $a^m b^n$, $m \neq n$. Therefore, non-members of L are accepted with probability at most $\frac{1}{N}$, and rejected with probability at least $1 - \frac{1}{N}$.

We now perform an analysis of the runtime. Stage 1 and Stage 2 take a total of at most $2|w| + 5$ steps. When $m \neq n$, the longest path followed in Stage 3 is the one with lowest speed on the more frequent input symbol. As seen in Eq. (3), this path is P_1 if $y < x$ (that is, $n < m$), and P_N if $x < y$. So, the worst-case runtime for Stage 3 when $m \neq n$ can be written as $|w| + xN + \max(1(y - x), N(y - x)) + x + 1$. Note that this bound remains valid when $m = n$ as well. Therefore, the overall running time of $M_{k,N}$ is at most $3|w| + 6 + \max(x, y)N + \min(x, y)$. When we set the divisor $k = N$, this value is bounded from above by $4|w| + 6$; therefore, the running time of $M_{N,N}$ is $O(|w|)$. Moreover, setting $N = \lceil \frac{1}{\epsilon} \rceil$ for any desired positive error bound ϵ, it is easily calculated that the size of the state set of $M_{N,N}$ is $O((\frac{1}{\epsilon})^2)$. □

One can reduce the constant hidden in the O-notation for the runtime to 2, at the cost of an increase in the corresponding constant for the state complexity, by modifying $M_{N,N}$ in the following way. We start by eliminating Stage 2, incorporating its functionality in Stage 3, which already involves modulo-N counting. Note that one has to add more reject states than one deletes ($N - 1$ new reject states for each of the N paths of Stage 3) in this transformation, since we no longer have the guarantee that every path P_j ends up at the state $m_{j,0}$ in this case. This increases the hidden constant in front of the state cost by 1. We then construct a new machine which makes a single pass of the tape, running this new version of Stage 3 in parallel with Stage 1, which results in that constant being multiplied anew by the (fixed) number of states of Stage 1.

I	Start as usual by checking whether w is of the form $a^m b^n$; $m, n > 0$. If not, REJECT. (Stage 1 from the previous section)
II	Zig–zag on the tape to perform r rounds of Stage 2 to check whether $m \equiv n \mod (k_i)$ in the i^{th} round. If any check fails, REJECT.
III	Perform r rounds of Stage 3. The i^{th} round branches to N_i paths and uses a divisor of value k_i. In all of these rounds (except the last), of the N_i targets of the QFT at the end, the distinguished target is not an accept state, but a non–halting state. Computation paths that arrive at this state split into N_{i+1}, and Stage 3 is performed again, this time with the head moving in the reverse direction and with the divisor set to k_{i+1}. The N_r–way QFT at the end of the last round has an accept state as its distinguished target, and the other targets are reject states.

Fig. 3. Description of the general multiple-pass construction.

5. State-efficient probability amplification using collision avoidance

The methods to be described in this section produce machines which are superior in both the description size and runtime measures to those of [10]. They involve the repeated application of the quotient comparison procedure in multiple passes over the input string, with the aim of reducing the error probability to the desired value. The divisors used in these passes are selected carefully to avoid unwanted interference between computation paths, which results in improved probability amplification.

The three constructions to be described here can be seen as different instantiations of the common template in Fig. 3, parameterized with two lists of integers, $K = (k_1, k_2, \ldots, k_r)$ and $B = (N_1, N_2, \ldots, N_r)$. (Consult Figs. 1 and 2 to recall the details of the stages of the basic quotient comparison procedure mentioned in the description given in Fig. 3.)

If the input to a machine of the type described in Fig. 3 is a member of L, then Stages I and II are passed without rejection. By the analysis in the proof of Proposition 1, all computation paths in the same round of Stage III reach the end-marker at the same step, and therefore the input is accepted with probability 1, regardless of the settings of K and B.

We will choose K and B to ensure that, if the input is not a member of L, and Stages I and II are passed without rejection, all pairs of distinct computation paths finish each round of Stage III at different times. (Recall from Sections 3.2 and 3.3 that the incorporation of this property was seen to reduce the state complexity of an alternative construction for recognizing L.) The details of how K and B may be selected will be discussed shortly, but first, let us examine the error bound achieved when the 2qfa is assumed to possess this "no-collision" guarantee: When such a machine runs on a string of the form $a^m b^n$ ($m \neq n$), there are $N_1 N_2 \cdots N_r$ distinct computation paths ending with accepting configurations. Since we know that no two paths merge, all we need to do in order to find the overall incorrect acceptance probability is to add the probabilities associated with these configurations. Each such accepting path undergoes $2r$ branchings. At the beginning of round l of Stage III, it selects one of the N_l branches of that round with probability $\frac{1}{N_l}$. In the QFT at the end of round l, the "lucky" branch to the distinguished target is selected with probability $\frac{1}{N_l}$. So the probability associated with the accepting configuration at the end of each such path is $\frac{1}{(N_1 N_2 \cdots N_r)^2}$. Multiplying this by the number of accepting paths, we see that the error bound ϵ equals $\frac{1}{N_1 N_2 \cdots N_r}$.

To start our analysis of how K and B may be chosen to avoid collisions, let P_i and P_j be two distinct computation paths of a machine fitting the general multiple-pass template of Fig. 3. P_i and P_j are distinguished by the branches that they follow in the splits at the beginning of each round of Stage III. Let us say that P_i and P_j select the $c_{l,i}$th and $c_{l,j}$th branches respectively in round l (where $1 \leq c_{l,i}, c_{l,j} \leq N_l$). The difference between the total lengths (i.e. the number of steps since the beginning of the computation) of P_i and P_j when they reach the end of round p can easily be found, making use of Eq. (3) from Section 4, to be

$$steps_i - steps_j = (n - m)\left(\frac{c_{1,i} - c_{1,j}}{k_1} + \frac{c_{2,i} - c_{2,j}}{k_2} + \cdots + \frac{c_{p,i} - c_{p,j}}{k_p}\right). \tag{4}$$

Since we are working on the case where $m \neq n$, in order to avoid unwanted interference, one has to select B and K so that the expression in the second set of parentheses above cannot equal 0 for any round p. Some of the terms in that set of parentheses may be 0, since some $c_{l,i}$ may equal the corresponding $c_{l,j}$, but this cannot be the case for all $(c_{l,i}, c_{l,j})$ pairs, since P_i and P_j are different computation paths. Concentrating on the non-zero terms, we see that the sum of all the positive terms must not equal the absolute value of the sum of all the negative terms for this expression to have a non-zero value. The following proposition will be used to ensure this condition for all the methods to be presented in this section.

Proposition 2. *Let $P = \{p_1, p_2, \ldots, p_s\}$ and $Q = \{q_1, q_2, \ldots, q_t\}$ be any two disjoint sets of prime numbers. Then $\sum_{i=1}^{s} \frac{c_i}{p_i} \neq \sum_{j=1}^{t} \frac{d_j}{q_j}$, whenever the c_i and the d_j are integers, and $0 < c_i < p_i$, $0 < d_j < q_j$.*

Proof. Define F, G, F_i, and G_j as follows ($1 \leq i \leq s$, $1 \leq j \leq t$):

$$F = p_1 p_2 \cdots p_s \quad \text{and} \quad F_i = \frac{F}{p_i},$$
$$G = q_1 q_2 \cdots q_t \quad \text{and} \quad G_j = \frac{G}{q_j}.$$

Suppose that $\sum_{i=1}^{s} \frac{c_i}{p_i} = \sum_{j=1}^{t} \frac{d_j}{q_j}$. Rewriting, one obtains

$$\left(\frac{c_1}{p_1} + \frac{c_2}{p_2} + \cdots + \frac{c_s}{p_s}\right) = \left(\frac{d_1}{q_1} + \frac{d_2}{q_2} + \cdots + \frac{d_t}{q_t}\right),$$
$$\left(\frac{c_1F_1 + c_2F_2 + \cdots + c_sF_s}{F}\right) = \left(\frac{d_1G_1 + d_2G_2 + \cdots + d_tG_t}{G}\right),$$

which yields

$$G(c_1F_1 + c_2F_2 + \cdots + c_sF_s) = F(d_1G_1 + d_2G_2 + \cdots + d_tG_t).$$

Dividing both sides by p_1, and rearranging, we get

$$\left(\frac{Gc_1F_1}{p_1}\right) = F_1(d_1G_1 + d_2G_2 + \cdots + d_tG_t) - G\left(\frac{c_2F_2 + \cdots + c_sF_s}{p_1}\right).$$

Since F_i ($2 \leq i \leq s$) has p_1 as a factor, the right-hand side of this equation is an integer. Since neither G nor F_1 have p_1 as a factor, and $0 < c_1 < p_1$, the left-hand side is not an integer, which is a contradiction. □

This gives us one way of guaranteeing that a multi-pass 2qfa matching the template of Fig. 3 avoids all unwanted collisions of computation paths. We select K such that all its elements are distinct prime numbers, and we make sure that no element N_i of B is greater than the corresponding element k_i of K. It is easy to see that Proposition 2 ensures that the path length difference of Eq. (4) will never equal zero when $m \neq n$ under these conditions.

K and B are set to values that satisfy the constraints described in the previous paragraph in our state-efficient constructions that will be presented in the following subsections.

5.1. N-way branching in a constant number of passes

Given a positive error bound ϵ and an integer $c > 1$, select K as the list of the first c primes greater than or equal to $(\frac{1}{\epsilon})^{(\frac{1}{c})}$. B is set to equal K.

The 2qfa's built according to this prescription traverse the tape $c+1$ times in Stages I and II. Since the number of branches N_i equals the divisor k_i in each round of Stage III, the runtime analysis of the basic machine $M_{N,N}$ from Section 4 applies, and the longest possible computation path, which includes the concatenation of the longest branches that can be followed in each of the c rounds of Stage III, has length $O(|w|)$, since c does not depend on ϵ.

As for the state complexity, recalling that the number of states of $M_{N,N}$ is $O(N^2)$, we see that the size of the multi-pass machine is bounded by the sum of the squares of the c primes mentioned above, which is easily found to be $O((\frac{1}{\epsilon})^{\frac{2}{c}})$, with the constant hidden in the big-O having a value on the order of $c^3 \log^2 c$.

5.2. N-way branching in multiple quotient comparison passes

For a given ϵ, select K (and B) as the list of the first r_1 primes, where r_1 is the smallest integer such that the product of all the elements of K is greater than or equal to $\frac{1}{\epsilon}$.

Using the well-known formula for the asymptotic growth of the primorial function (the product of primes less than or equal to the r_1th prime), that is,

$$\prod_{i=1}^{r_1} p_i \sim \mathrm{e}^{(1+o(1))r_1 \log r_1}, \tag{5}$$

one can calculate

$$r_1 = O\left(\frac{\log \frac{1}{\epsilon}}{\log \log \frac{1}{\epsilon}}\right) \tag{6}$$

for the smallest integer value of r_1. Since r_1 passes of stages of machines of type $M_{N,N}$ are performed in each of Stages II and III, this method yields machines with runtime $O(\frac{\log \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}}|w|)$, which is still faster than what one would obtain by applying the conventional repetitive probability amplification method to a linear-time classical machine. The state complexity is proportional to the sum of the squares of the first r_1 primes, that is,

$$\sum_{i=1}^{r_1} p_i^2 \sim \frac{1}{3} r_1^3 \log^2 r_1, \tag{7}$$

and turns out to be

$$O\left(\frac{\log^3 \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}}\right), \tag{8}$$

which is a major improvement over that of the machines of Section 5.1.

5.3. Two-way branching in multiple quotient comparison passes

For a given ϵ, select K as the list of the first r_2 primes, where r_2 is the smallest integer such that $2^{r_2} \geq \frac{1}{\epsilon}$. Set all r_2 elements of B to 2.

$\lceil \log \frac{1}{\epsilon} \rceil$ passes are performed in each of Stages II and III. Since the machine $M_{k,N}$ of Section 4 has time complexity $O(|w|)$ when N is held constant, independent of ϵ, this method yields machines with runtime $O(\log(\frac{1}{\epsilon})|w|)$. The state complexity, which is mainly due to the modulo-k_i counting in this case, can be calculated asymptotically by using the well-known formula for the sum of the first r_2 primes, namely,

$$\sum_{i=1}^{r_2} p_i \sim \frac{1}{2} r_2^2 \ln r_2, \tag{9}$$

and turns out to be $O(\log^2(\frac{1}{\epsilon}) \log\log \frac{1}{\epsilon})$, which is better than what we had in Section 5.2.

6. Concluding remarks

Watrous [14] notes that the remarkable inefficiency encountered when one attempts to apply the conventional repetition-based probability amplification technique to 2qfa's stems from the fact that their definition does not allow these machines to "reset" themselves when running, and suggests that a modified model of automata, with classical as well as quantum parts, could be defined to avoid this problem. (Indeed, Watrous, in collaboration with Andris Ambainis, later developed a model of two-way finite automata with quantum and classical states (2qcfa's), which we discuss below.) In this paper, we demonstrated that highly efficient probability amplification is possible within the 2qfa framework, focusing on the case of L, arguably the most "famous" language known to be decidable by these machines. It would certainly be interesting to examine how well our techniques can be ported to other 2qfa's for other languages. The idea of "collision avoidance," that is, concatenating slower and slower versions, rather than identical copies, of the basic machine whose error is to be reduced (Sections 3.3 and 5), seems to be promising for QFT-based algorithms in general. Other probability amplification methods, making novel use of the quantum nature of the model, could also exist.

It has been noted [2] that the 2qfa model is difficult to implement, since the number of quantum states necessary to keep track of the head positions grows as the input string gets longer. In 2qcfa's, the head position is classical, and the size of the quantum part is constant. Ambainis and Watrous have shown [3] that this weaker setup can be used to build machines that recognize L with constant state set size, where the "tuning" of the automaton for a particular error bound is achieved by setting some transition amplitudes appropriately, a very efficient method of probability amplification indeed. On the other hand, the expected runtime of these machines is $O(|w|^4)$. (It should be stressed that these 2qcfa's for L have no upper bound on their worst-case runtime, whereas all computation paths of all the 2qfa's we have described halt within the time bound indicated in Table 1.)

If one is interested in minimizing the state complexity of the resulting 2qfa's at the expense of increasing the time complexity to a level beyond $O(\log(\frac{1}{\epsilon})|w|)$ (the cost one would incur in classical probability amplification), one can attempt to adapt the philosophy of the Ambainis–Watrous 2qcfa construction for L to the 2qfa setup, by essentially rearranging a small machine of the type discussed in Section 3.1 to run in an infinite loop, in which the QFT's targets are reject and non-halting states, and adding a subroutine which accepts with a small probability in each iteration. This approach is complicated by the fact that computation paths of different "rounds" would now collide, a problem that can be mitigated by periodically rotating the complex amplitudes of these paths to reduce the unwanted interference. We do not dwell further on this construction, whose expected runtime would necessarily be worse than the worst runtime in Table 1.

Table 1 contains a quick comparison of our methods with previous work. The runtimes of Methods 4 and 5, in which the tape is traversed a fixed number of times, independent of the desired error bound, are clearly optimal. The three "best" methods (5, 6, and 7) exhibit an interesting trade-off between time and state costs. It is an open question whether this is an inherent feature of the problem, or the combined runtime-state complexities reported here can be improved even further.

Table 1
Comparisons of methods for recognizing L w.r.t. runtime and size of state set.

#	Described in Section #	Runtime	Size of state set
1	3.1	$O(\frac{1}{\epsilon}\|w\|)$	$O((\frac{1}{\epsilon})^2)$
2	3.2	$O((\frac{1}{\epsilon})^2\|w\|)$	$O((\frac{1}{\epsilon})^2)$
3	3.3	$O(\frac{1}{\epsilon}\|w\|)$	$O(\frac{1}{\epsilon})$
4	4	$O(\|w\|)$	$O((\frac{1}{\epsilon})^2)$
5	5.1	$O(\|w\|)$	$O((\frac{1}{\epsilon})^{\frac{2}{\epsilon}})$
6	5.2	$O(\frac{\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}}\|w\|)$	$O(\frac{\log^3\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}})$
7	5.3	$O(\log(\frac{1}{\epsilon})\|w\|)$	$O(\log^2(\frac{1}{\epsilon})\log\log\frac{1}{\epsilon})$

Acknowledgements

We thank Cem Yalçın Yıldırım for his helpful answers to our mathematical questions. We are grateful to the anonymous reviewer, whose comments helped us improve the exposition substantially. This research was partially supported by the Boğaziçi University Research Fund with grant 07A103.

References

[1] A. Ambainis, A. Ķikusts, M. Valdats, On the class of languages recognizable by 1-way quantum finite automata, in: A. Ferreira, H. Reichel (Eds.), STACS 2001: 18th Annual Symposium on Theoretical Aspects of Computer Science, in: Lecture Notes in Computer Science, vol. 2010, Springer, 2001.
[2] A. Ambainis, R. Freivalds, 1-way quantum finite automata: Strengths, weaknesses and generalizations, in: FOCS'98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, Palo Alto, California, 1998.
[3] A. Ambainis, J. Watrous, Two-way finite automata with quantum and classical states, Theoretical Computer Science 287 (1) (2002) 299–311.
[4] F.M. Atak, Design and simulation of two-way quantum finite automata, Master's thesis, Boğaziçi University, İstanbul, Turkey, 2006.
[5] A. Bertoni, M. Carpentieri, Regular languages accepted by quantum automata, Information and Computation 165 (2) (2001) 174–182.
[6] A. Brodsky, N. Pippenger, Characterizations of 1-way quantum finite automata, SIAM Journal on Computing 31 (5) (2002) 1456–1478.
[7] C. Dwork, L.J. Stockmeyer, On the power of 2-way probabilistic finite state automata (extended abstract), in: FOCS'89: Proceedings of the 30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, 1989.
[8] R. Freivalds, Probabilistic two-way machines, in: J. Gruska, M. Chytil (Eds.), Proceedings of the International Symposium on Mathematical Foundations of Computer Science, in: Lecture Notes in Computer Science, vol. 118, Springer, 1981.
[9] J. Gruska, Descriptional complexity issues in quantum computing, Journal of Automata, Languages and Combinatorics 5 (3) (2000) 191–218.
[10] A. Kondacs, J. Watrous, On the power of quantum finite state automata, in: FOCS'97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science, Miami, Florida, 1997.
[11] C. Moore, J.P. Crutchfield, Quantum automata and quantum grammars, Theoretical Computer Science 237 (1–2) (2000) 275–306.
[12] D. Qiu, M. Ying, Characterizations of quantum automata, Theoretical Computer Science 312 (2–3) (2004) 479–489.
[13] J.C. Shepherdson, The reduction of two-way automata to one-way automata, IBM Journal of Research and Development 3 (1959) 198–200.
[14] J. Watrous, On the power of 2-way quantum finite state automata, Tech. Rep. CS-TR-1997-1350, University of Wisconsin, 1997. URL: citeseer.ist.psu.edu/article/watrous97power.html.